# EXHIBIT K

**U.S. Patent No. 10,375,215 – Google LLC**
**Claim 1**

Jenam Tech LLC ("Jenam") provides evidence of infringement of Claim 1 of U.S. Patent No. 10,375,215 (hereinafter "the '215 patent") by Google LLC ("Google"). In support thereof, Jenam provides the following claim charts.

"Accused Instrumentalities" as used herein refers to at least one or more websites or web addresses including, but not limited to www.google.com, stored and/or hosted on one or more servers owned or under the control of Google. These claim charts demonstrate Google's infringement, and provide notice of such infringement, by comparing each element of the asserted claims to corresponding components, aspects, and/or features of the Accused Instrumentalities. These claim charts are not intended to constitute an expert report on infringement. These claim charts include information provided by way of example, and not by way of limitation.

The analysis set forth below is based only upon information from publicly available resources regarding the Infringing Instrumentalities, as Google has not yet provided any non-public information. An analysis of Google's (or other third parties') technical documentation and/or software source code may assist in fully identify all infringing features and functionality. Accordingly, Jenam reserves the right to supplement this infringement analysis once such information is made available to Jenam. Furthermore, Jenam reserves the right to revise this infringement analysis, as appropriate, upon issuance of a court order construing any terms recited in the asserted claims.

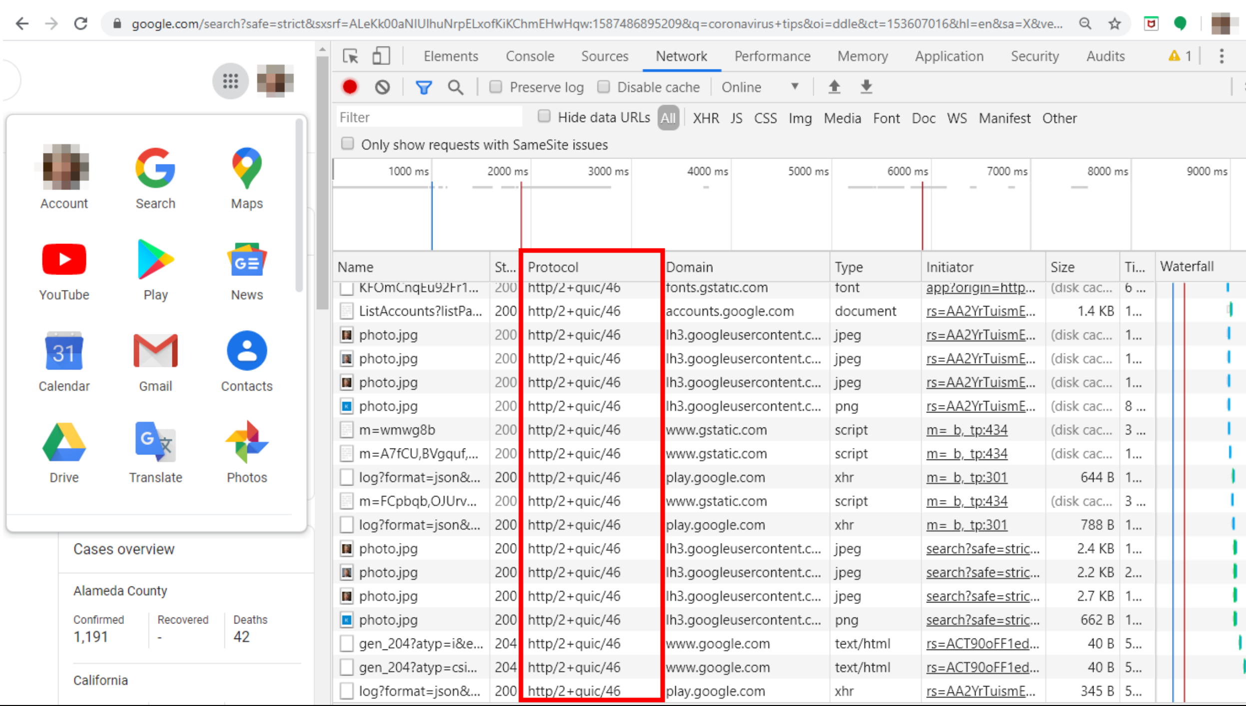
Unless otherwise noted, Jenam contends that Google directly infringes the '215 patent in violation of 35 U.S.C. § 271(a) by selling, offering to sell, making, using, and/or importing the Infringing Instrumentalities. The following exemplary analysis demonstrates that infringement. Unless otherwise noted, Jenam further contends that the evidence below supports a finding of indirect infringement under 35 U.S.C. §§ 271(b) and/or (c), in conjunction with other evidence of liability under one or more of those subsections. Google makes, uses, sells, imports, or offers for sale in the United States, or has made, used, sold, imported, or offered for sale in the past, without authority, or induces others to make, use, sell, import, or offer for sale in the United States, or has induced others to make, use, sell, import, or offer for sale in the past, without authority products, equipment, or services that infringe Claim 1 of the '215 patent, including without limitation, the Accused Instrumentalities.

Unless otherwise noted, Jenam believes and contends that each element of each claim asserted herein is literally met through Google's provision of the Infringing Instrumentalities. However, to the extent that Google attempts to allege that any asserted claim element is not literally met, Jenam believes and contends that such elements are met under the doctrine of equivalents. More specifically, in its investigation and analysis of the Infringing Instrumentalities, Jenam did not identify any substantial differences between the elements of the patent claims and the corresponding features of the Infringing Instrumentalities, as set forth herein. In each instance, the identified feature of the Infringing Instrumentalities performs at least substantially the same function in substantially the same way to achieve substantially the same result as the corresponding claim element.

To the extent the chart of an asserted claim relies on evidence about certain specifically-identified Accused Instrumentalities, Jenam asserts that, on information and belief, any similarly-functioning instrumentalities also infringes the charted claim. Jenam reserves the right to amend this infringement analysis based on other products made, used, sold, imported, or offered for sale by Google. Jenam also reserves the right to amend this infringement analysis by citing other claims of the '215 patent, not listed in the claim chart, that are infringed by the Accused Instrumentalities. Jenam further reserves the right to amend this infringement analysis by adding, subtracting, or otherwise modifying content in the "Accused Instrumentalities" column of each chart.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,375,215

| Claim 1 | Accused Instrumentalities |
|---|---|
| A computer-implemented method, comprising: at a server node: causing data to be sent to a client node; and providing access, to the client node, to code that, in response to being used by the client node, causes the client node to:<br><br>receive, by the client node, a transmission control protocol (TCP)-variant packet, | Google owns or controls a server node that performs a method including: at the server node: causing data (e.g., in HTML pages, etc.) to be sent to a client node (e.g., device that receives the HTML pages, etc.); and providing access, to the client node, to code (e.g., also in the HTML pages, etc.) that, in response to being used by the client node, causes the client node to carry out the following functionality. Specifically, Google incorporates code into their HTML pages that, when used by the client node, causes the client node to communicate with one or more servers using the QUIC protocol.<br><br>See excerpt(s) below, for example (emphasis added, if any):<br><br>**Note**: Below is a web page of Google ([https://www.google.com/](https://www.google.com/)) . |






June 1, 2020

| | |
|---|---|
| | **Note**: As set forth below, QUIC is a "variant" of TCP.<br><br>1. Introduction<br><br>QUIC is a multiplexed and secure transport protocol that runs on top of UDP. QUIC aims to provide a flexible set of features that allow it to be a general-purpose transport for multiple applications.<br><br>QUIC implements techniques learned from experience with TCP, SCTP and<br><br>"On the surface, QUIC is very similar to TCP+TLS+HTTP/2 implemented on UDP. ...However, since QUIC is built on top of UDP, it suffers from no such limitations." https://www.chromium.org/quic<br><br>**Note**: As set forth below, a QUIC negotiation packet is received by the client node from a server node.<br><br>7.4. Transport Parameters<br><br>During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling.<br><br>QUIC packet: A well-formed UDP payload that can be parsed by a QUIC receiver. QUIC packet size in this document refers to the UDP<br><br>**Note**: As set forth below, prior to a QUIC connection being established, the QUIC connection is "set up" using the aforementioned handshake. |

| | |
|---|---|
| | 3.1. Low-Latency Connection Establishment<br><br>QUIC relies on a combined cryptographic and transport handshake for setting up a secure transport connection. QUIC connections are expected to commonly use 0-RTT handshakes, meaning that for most QUIC connections, data can be sent immediately following the client handshake packet, without waiting for a reply from the server. QUIC provides a dedicated stream (Stream ID 0) to be used for performing the cryptographic handshake and QUIC options negotiation. The format of the QUIC options and parameters used during negotiation are described in this document, but the handshake protocol that runs on Stream ID 0 is described in the accompanying cryptographic handshake draft [QUIC-TLS]. |
| detect an idle time period parameter field in the TCP-variant packet; | Google owns or controls the server node that performs the method including: at the server node: providing access, to the client node, to the code (e.g., in the HTML pages, etc.) that, in response to being used by the client node, causes the client node to: detect an idle time period parameter field (e.g., idle timeout parameter field, etc.) in the TCP-variant packet (e.g., QUIC negotiation packet, etc.).<br><br>See excerpt(s) below, for example (emphasis added, if any):<br><br>**<u>Note</u>**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet.<br><br>7.4. Transport Parameters<br><br>During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. |

| | |
|---|---|
| | (see nested table below) |

| Value | Parameter Name | Specification |
|---|---|---|
| 0x0000 | initial_max_stream_data | Section 7.4.1 |
| 0x0001 | initial_max_data | Section 7.4.1 |
| 0x0002 | initial_max_stream_id | Section 7.4.1 |
| 0x0003 | idle_timeout | Section 7.4.1 |
| 0x0004 | omit_connection_id | Section 7.4.1 |
| 0x0005 | max_packet_size | Section 7.4.1 |
| 0x0006 | stateless_reset_token | Section 7.4.1 |

Table 4: Initial QUIC Transport Parameters Entries

| | |
|---|---|
| | The format of the transport parameters is the TransportParameters struct from Figure 6. This is described using the presentation language from Section 3 of [I-D.ietf-tls-tls13].<br><br>`uint32 QuicVersion;`<br><br>`enum {`<br>`initial_max_stream_data(0),`<br>`initial_max_data(1),`<br>`initial_max_stream_id(2),`<br>`idle_timeout(3),`<br>`omit_connection_id(4),`<br>`max_packet_size(5),`<br>`stateless_reset_token(6),`<br>`(65535)`<br>`} TransportParameterId;` |
| identify metadata in the idle time period parameter field for an idle time period, during which, no packet is communicated in a TCP-variant connection to keep the TCP-variant connection active; and | Google owns or controls the server node that performs the method including: at the server node: providing access, to the client node, to the code (e.g., in the HTML pages, etc.) that, in response to being used by the client node, causes the client node to: identify metadata (e.g., a value, etc.) in the idle time period parameter field (e.g., idle timeout parameter field, etc.) for an idle time period, during which, no packet is communicated in a TCP-variant connection to keep the TCP-variant connection active.<br><br>See excerpt(s) below, for example (emphasis added, if any):<br><br>**<u>Note</u>**: As set forth below, the metadata includes a value in seconds that is encoded as an unsigned 16-bit integer.<br><br>`idle_timeout (0x0003): The idle timeout is a value in seconds that is encoded as an unsigned 16-bit integer. The maximum value is 600 seconds (10 minutes).`<br><br>**<u>Note</u>**: As set forth below, there is "no activity" while the connection is idle. |

| | |
|---|---|
| | 7.8.2. Idle Timeout<br><br>A connection that remains idle for longer than the idle timeout (see Section 7.4.1) becomes closed. Either peer removes connection state if they have neither sent nor received a packet for this time.<br><br>The time at which an idle timeout takes effect won't be perfectly synchronized on peers. A connection enters the draining period when the idle timeout expires. During this time, an endpoint that receives new packets MAY choose to restore the connection. Alternatively, an endpoint that receives packets MAY signal the timeout using an immediate close. |
| determine, based on the metadata, a timeout attribute associated with the TCP-variant connection. | Google owns or controls the server node that performs the method including: at the server node: providing access, to the client node, to the code (e.g., in the HTML pages, etc.) that, in response to being used by the client node, causes the client node to: determine, based on the metadata (e.g., the value of the idle timeout parameter field, etc.), a timeout attribute associated with the TCP-variant connection.<br><br>See excerpt(s) below, for example (emphasis added, if any):<br><br>**Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet.<br><br>idle_timeout (0x0003): The idle timeout is a value in seconds that is encoded as an unsigned 16-bit integer. The maximum value is 600 seconds (10 minutes).<br><br>7.8. Connection Termination<br><br>Connections should remain open until they become idle for a pre-negotiated period of time. A QUIC connection, once established, can be terminated in one of three ways: |

| | |
|---|---|
| | 7.8.2. Idle Timeout<br><br>A connection that remains idle for longer than the idle timeout (see Section 7.4.1) becomes closed. Either peer removes connection state if they have neither sent nor received a packet for this time.<br><br>The time at which an idle timeout takes effect won't be perfectly synchronized on peers. A connection enters the draining period when the idle timeout expires. During this time, an endpoint that receives new packets MAY choose to restore the connection. Alternatively, an endpoint that receives packets MAY signal the timeout using an immediate close. |

**Caveat**: The notes and/or cited excerpts utilized herein are set forth for illustrative purposes only and are not meant to be limiting in any manner. For example, the notes and/or cited excerpts, may or may not be supplemented or substituted with different excerpt(s) of the relevant reference(s), as appropriate. Further, to the extent any error(s) and/or omission(s) exist herein, all rights are reserved to correct the same in connection with any subsequent correlations.